Procédure DEBUT

1 But

Affecter les ressources mémoire, disque et fichiers.

L'exécution est constituée d'un ensemble de commandes commençant par DEBUT et se terminant par FIN [U4.11.02], (voir aussi la procédure POURSUITE [U4.11.03]).

La commande DEBUT qui est exécutée, dès sa lecture par le Superviseur, effectue les tâches suivantes :

- définition des caractéristiques des bases de données (gérées par JEVEUX) et allocation des fichiers associés,
- lecture des catalogues des éléments et des commandes.

La syntaxe apparemment complexe de cette procédure ne doit pas inquiéter l'utilisateur ; son appel avec les opérandes par défaut, suffisant dans la plupart des cas, est : DEBUT ( )

Les opérandes sont à utiliser dans le cas d'études nécessitant une taille des fichiers "bases de données" plus importante ou pour dérouter les différents fichiers sur des numéros d'unité logique différents des numéros affectés par défaut.

Les commandes placées avant DEBUT, si elles sont syntaxiquement correctes, sont ignorées.

Table des matières

2 Syntaxe

```
DEBUT
        (  ◊   PAR_LOT       =  /   'OUI',                         [DEFAUT]
                                /   'NON',
           ◊   IMPR_MACRO    =  /   'NON',                         [DEFAUT]
                                /   'OUI',

           ◊   LANG = lang,                                        [TXM]

           ◊   BASE = _F (      ♦   FICHIER = /   'GLOBALE',
                                              /   'VOLATILE',
                                              /   |LONG_ENRE = lenr,  [I]
                                                  |NMAX_ENRE = nenr,  [I]
                                                  |LONG_REPE = lrep,  [I]
                         ),

           ◊   CODE = _F  (  ♦   NOM =     nom code,               [K8]
                             ♦   NIV_PUB_WEB     =  / 'INTERNET' ,
                                                    / 'INTRANET' ,
                             ◊   VISU_EFICAS     =  /   'OUI',     [DEFAUT]
                                                    /   'NON',
                             ),

           ◊   ERREUR =  _F  (ERREUR_F=       /   'ABORT',         [DEFAUT]
                                                  /   'EXCEPTION',
                                ),
           ◊   IGNORE_ALARM =   l_vale,                            [l_Kn]

           ◊   DEBUG =       _F  (  JXVERI =  /   'OUI',
                                              /   'NON',
                                    ENVIMA =      'TEST',          [l_Kn]
                                ◊   JEVEUX =  /   'OUI',
                                              /   'NON',
                                ◊   SDVERI =  /   'OUI',
                                                  'NON',
                                ◊   HIST_ETAPE = /   'NON',
                                                     'OUI',
                                    ),
           ◊   MESURE_TEMPS = _F(
                                ◊   NIVE_DETAIL =/ 0               [DEFAUT]
                                                 / 1
                                                 / 2
                                                 / 3
                                ◊   MOYENNE =    / 'OUI'           [DEFAUT]
                                                 / 'NON'
                                ),
           ◊   MEMOIRE = _F(◊   TAILLE_BLOC     =  /   800.,       [DEFAUT]
                                                   /   tbloc,      [R]
                         ),

           ◊   CATALOGUE  = _F( ♦ FICHIER     = nfic,              [l_Kn]
                                ◊ UNITE       = unite,             [I]
                             ),

           ◊   RESERVE_CPU =_F(    / VALE        =  vale           [R]
                                   / POURCENTAGE= pcent            [R]

                                   ◊ BORNE    =  /   bv,           [R]
                                                 /   180.        [DEFAUT]
                              ),
```

)

3 Affichages

Au début de l'exécution de Code_Aster, un entête est affiché. On y trouve :

- l'identification précise de la version utilisée : numéro de version, date des dernières modifications,
- la date et l'heure du début de l'exécution,
- le nom, l'architecture, le système d'exploitation de la machine,
- la langue utilisée pour l'affichage des messages,
- le type de parallélisme disponible (MPI/OpenMP), le nombre de processeurs alloués,
- la version des bibliothèques utilisées (quand elle est disponible) pour hdf5, med, mumps, scotch,
- puis plusieurs informations sur la répartition de la mémoire.
  Par exemple :

```
Mémoire limite pour l'exécution : 256.00 Mo
consommée par l'initialisation : 148.68 Mo
par les objets du jeu de commandes : 17.48 Mo
reste pour l'allocation dynamique : 89.84 Mo
Taille limite des fichiers d'échange : 48.00 Go
```

Ce qui signifie :

- `256 Mo` est la quantité de mémoire demandée par l'utilisateur, c'est la quantité totale qu'il ne faut pas dépasser.
- `148.68 Mo` est consommé simplement en démarrant l'exécution (chargement de l'exécutable, des bibliothèques dynamiques associées, etc.).
- `17.48 Mo` est consommé par la lecture du fichier de commandes (Remarque : en mode PAR_LOT='NON', le jeu de commandes étant lu au fur et à mesure, cette valeur sera alors nulle).
- `89.84 Mo` est la quantité de mémoire disponible (à cet instant) pour les objets du calcul (égale à 256-148.68-17.48). On voit donc que le calcul ne peut pas débuter si cette valeur est trop faible.

Au cours de l'exécution, en fonction des allocations dynamiques effectuées, lorsque cette valeur varie de plus de 10 % (à la hausse ou à la baisse), un message de ce type informe l'utilisateur :

```
La mémoire consommée actuellement hors JEVEUX est de 214.08 Mo.
La limite de l'allocation dynamique JEVEUX est fixée à 41.92 Mo.
```

En fin d'exécution, un bilan indique si le même calcul peut être relancé avec moins de mémoire :

```
La mémoire demandée au lancement est surestimée, elle est de 256 Mo.
Le pic mémoire utilisée est de 216.02 Mo.
```

ou si plus de mémoire est nécessaire (en effet selon les plates-formes, la limite maximum peut être dépassé sans que le système ait interrompu le calcul) :

```
La mémoire demandée au lancement est sous-estimée, elle est de 256 Mo.
Le pic mémoire utilisée est de 273.22 Mo.
```

4 Opérandes

4.1 Opérande `PAR_LOT`

`PAR_LOT =`

Mode de traitement des commandes :

`'OUI'`: (option par défaut) ; le superviseur analyse **toutes** les commandes avant d'en demander l'exécution.

`'NON'`: après avoir analysé **une** commande le superviseur demande son exécution puis passe à l'analyse (et à l'exécution) de la commande suivante (traitement commande par commande).

4.2 Mot clé `IMPR_MACRO`

`IMPR_MACRO =`

Autorise ou non les affichages produits par les macros dans le fichier de message. La lecture des fichiers de message peut être pénible quand elle contient la totalité des échos des sous-commandes

générées par la macro elle-même. Par défaut, seul l'écho des commandes explicitement appelées par l'utilisateur dans son jeu de commandes apparaîtra.

4.3 Mot-clé LANG

Il permet de choisir la langue d'affichage des messages émis par le code.
Si le mot-clé n'est pas renseigné, ce sont les variables d'environnement qui détermine la langue des messages (référence : http://www.gnu.org/software/gettext/manual/gettext.html#Users). On peut par exemple définir dans le fichier `~/.bashrc` : `export LANG=fr_FR.UTF-8`.
L'encodage (`UTF-8` ou `ISO-8859-1`) permet d'afficher correctement les caractères accentués.
Le mot-clé `LANG` attend un valeur en deux lettres, par exemple `'FR'` (pour le français) ou `'EN'` (pour l'anglais).
Quand une langue est choisie (que ce soit par l'environnement ou `LANG`), encore faut-il que le fichier des messages traduits (fichier `.mo`) soit disponible. Ce fichier est attendu sous ce nom :

```
$ASTER_ROOT/share/locale/`lang`/LC_MESSAGES/aster_`version`.mo
```

où `$ASTER_ROOT` est le répertoire principal de *Code_Aster* (ex. : `/aster` ou `/opt/aster`), `lang` est le nom en minuscules de la langue (ex. `en`, `fr`, `de`...) et `version` est le nom de la version de *Code_Aster* utilisée (ex. `stable`, `testing`, `unstable`).
Si le fichier de traduction ne peut être lu, c'est le français qui est utilisé.

Remarque

Même si le fichier de traduction existe, quand un message n'a pas été traduit, il est affiché en français (langue de rédaction des messages dans le code source).

4.4 Mot clé BASE

`BASE =`

La fonctionnalité de ce mot clé est de redéfinir les valeurs des paramètres des fichiers d'accès direct associés aux "bases de données" dans le cas où l'on ne désire pas utiliser ceux fixés par défaut.

Valeurs par défaut des paramètres associés aux bases de données.

GLOBALE		
NMAX_ENRE	62914	
LONG_ENRE	100	Kmots
LONG_REPE	2000	
VOLATILE		
NMAX_ENRE	62914	
LONG_ENRE	100	Kmots
LONG_REPE	2000	

Le mot vaut 8 octets sur plate-forme 64 bits sous LINUX 64, TRU64 et IRIX 64, 4 octets sur plate-forme 32 bits sous SOLARIS, HP-UX et WINDOWS-NT, LINUX.

Sous Linux 64, avec les valeurs par défaut, la procédure `DEBUT` allouera un fichier d'accès direct d'au plus `62914` enregistrements de `100 Kmots` (le `K` vaut `1024`) pour la base `'GLOBALE'`.

Remarque :

La taille réelle du fichier est dynamique ; elle dépend du volume d'informations à stocker effectivement. Mais cette taille est limitée par les conditions d'exploitation et un paramètre défini parmi les valeurs caractérisant la plate-forme. Sur la plate-forme de référence Linux 64 bits, la taille maximum est fixée à 48 Go. Cette valeur peut être modifiée en passant un argument sur la ligne de commande de l'exécutable derrière le mot clé « -max_base taille » où taille est une valeur réelle mesurée en Mo.

Sur les plates-formes 32 bits, la taille maximum est fixée à 2.047 Go (2 147 483 647), mais le code gère plusieurs fichiers pour aller au delà de cette limite lorsque le paramètre «-max_base » est passé en argument.
Pour la base Globale, qui peut être sauvegardée et ré-utilisée en donnée d'un calcul, la taille maximum en « POURSUITE » est conservée telle quelle si le paramètre « -max_base » n'est pas utilisé, mais peut-être redéfini au besoin de cette manière.

4.4.1 Opérande `FICHIER`

♦ `FICHIER =`

Nom symbolique de la base considérée.

4.4.2 Opérandes `LONG_ENRE / NMAX_ENRE / LONG_REPE`

Définition des paramètres de la base de données (fichiers d'accès direct).

`/ | LONG_ENRE = lenr`

`lenr` est la longueur des enregistrements en Kmots des fichiers d'accès directs utilisés.

Remarque :

Le gestionnaire de mémoire JEVEUX utilise ce paramètre pour déterminer deux types d'objets : les gros objets qui seront découpés en autant d'enregistrements que nécessaire, et les petits objets qui seront accumulés dans un tampon de la taille d'un enregistrement avant d'être déchargé.

`| NMAX_ENRE = nenr`

`nenr` est le nombre d'enregistrements par défaut, cette valeur est déterminée à partir de `LONG_ENRE` et d'un paramètre d'exploitation sur la plate-forme de référence Linux 64 fixé à 48 Go (51 539 607 552 octets) pour la taille maximale du fichier associé à une base de données, si cette valeur n'a pas été modifiée par l'utilisation du mot-clé –max_base sur la ligne de commande de l'exécutable.

Remarque :

Les deux opérandes `LONG_ENRE` et `NMAX_ENRE` doivent être utilisés avec précaution, un mauvais usage pouvant conduire à l'arrêt brutal du programme par saturation des fichiers d'accès direct. La cohérence entre la taille maximale du fichier et la valeur résultant du produit des deux paramètres `LONG_ENRE` et `NMAX _ENRE` est vérifiée en début d'exécution.

`| LONG_REPE = lrep`

`lrep` est la longueur initiale du répertoire (nombre maximal d'objets adressables par JEVEUX), elle est gérée dynamiquement par le gestionnaire de mémoire qui étend la taille du répertoire et de tous les objets système associés au fur et à mesure des besoins.

Remarque :

Le choix par l'utilisateur de modifier ces différents paramètres détermine de façon définitive certaines caractéristiques de la base `GLOBALE` qui ne peuvent plus être modifiées en `POURSUITE` .

4.5 Mot clé `CODE`

`CODE =`

Définition d'un nom pour l'ensemble d'une étude. Ce mot clé est destiné uniquement aux fichiers de commandes des tests de non régression gérés avec le code source.
La présence de ce mot clé déclenche l'émission d'un message d'information et positionne automatiquement le mode de deboggage `DEBUG ( JXVERI = 'OUI', )` qui met en oeuvre des vérifications sur les objets JEVEUX, ce qui peut amener un surcoût à l'exécution. Le comportement en cas d'erreur peut être modifié.

4.5.1 Opérande `NOM`

♦ NOM = nom code

Nom d'identification de l'étude, ce nom est au plus de 8 caractères.

4.5.2 Opérande NIV_PUB_WEB

♦ NIV_PUB_WEB = 'INTRANET'

Indicateur de niveau de publication. Signifiant que le test est uniquement diffusable sur le réseau interne.

NIV_PUB_WEB = 'INTERNET'

Indique que le test est diffusable tel quel sur le réseau externe.

VISU_EFICAS = 'OUI'

Indique que le fichier de commandes peut être ouvert sans problème avec l'outil EFICAS. Ce mot-clé est essentiellement utilisé pour les tests et à des fins de recette des nouvelles versions de l'outil.

VISU_EFICAS = 'NON'

Signale la présence de source python dans le fichier de commandes ne permettant pas son édition avec l'outil EFICAS.

4.6 Mot clé ERREUR

Permet de modifier le comportement du code en cas d'erreur <F>.

4.6.1 Opérande ERREUR_F

En cas d'erreur, le code interrompt l'exécution normale du jeu de commandes.
Par défaut, une exception est alors levée (pour la définition détaillée d'une exception Python, on se reportera à la documentation de Python ou à celle du superviseur, *cf.* [U1.03.01]). Dans ce cas, le code exécute la commande FIN (*cf.* [U4.11.02]) qui ferme alors la base afin de permettre la poursuite éventuelle du calcul. On remarquera que, bien que l'erreur initiale soit dite « fatale » (<F>), le diagnostic est <S>_ERROR puisque l'exception est « récupérée » par FIN. Cette base sera ensuite recopiée par le gestionnaire d'études. Ceci est le comportement quand ERREUR_F='EXCEPTION'.

Si ERREUR_F='ABORT', cela signifie qu'on demande explicitement au code d'interrompre définitivement l'exécution du jeu de commandes en cas d'erreur fatale (<F>). La commande FIN n'est pas exécutée, la base n'est donc pas fermée correctement, elle n'est pas recopiée et aucune reprise du calcul n'est possible.

Remarques

Pour l'exécution des cas-tests par les développeurs, l'arrêt par ABORT est automatique et par défaut. Ceci est activé par la présence du mot-clé facteur CODE (sauf si ERREUR_F précise autre chose).

En cas de manque de temps CPU, de mémoire, pour toutes erreurs de type <S> et les exceptions, le comportement est celui décrit quand ERREUR_F='EXCEPTION' .

4.7 Mot clé IGNORE_ALARM

IGNORE_ALARM =

Permet à l'utilisateur de supprimer l'affichage de certaines alarmes (dont il connaît l'origine) afin d'identifier plus facilement les autres alarmes qui pourraient apparaître.
Lors de l'exécution de la commande FIN, on affiche systématiquement un tableau récapitulatif des alarmes émises pendant l'exécution (et le nombre d'occurrences). Les alarmes ignorées par l'utilisateur sont précédées de '*' pour les distinguer (et elles apparaissent même si elles n'ont pas été émises).

Les alarmes sont désignées à partir de la nomenclature figurant entre les caractères < et >, par exemple : `IGNORE_ALARME = ('MED_2', 'SUPERVIS_40',...)`.

4.8 Mot clé DEBUG

`DEBUG =`

Option de déboggage (réservée aux développeurs et à la maintenance du code).

4.8.1 Opérande JXVERI

`JXVERI =`

Permet de contrôler l'intégrité des segments de la mémoire entre deux exécutions de commandes consécutives. Par défaut l'exécution s'effectue sans "DEBUG". Cette option est systématiquement activée en présence du mot clé CODE.

4.8.2 Opérande ENVIMA

`ENVIMA = 'TEST'`

Permet d'imprimer dans le fichier RESULTAT les valeurs des paramètres définis dans le progiciel ENVIMA caractérisant la machine [D6.01.01].

4.8.3 Opérande JEVEUX

◊JEVEUX =

Permet d'activer le mode de fonctionnement en debug du gestionnaire de mémoire JEVEUX : déchargements sur disque non différés et affectation des segments valeurs à une valeur indéfinie [D6.02.01].

4.8.4 Opérande SDVERI

`SDVERI = 'NON'`

L'usage de ce mot clé est à destination des développeurs. Attention, cette fonctionnalité peu provoquer un surcoût non négligeable lors de l'exécution.
Ce mot clé déclenche la vérification des structures de données produites par les opérateurs. Il est utilisé dans le cadre des procédures de développement du code dans les tests de non régression. Si le mot clé CODE est présent, ce mot clé prend la valeur par défaut 'OUI'.

4.8.5 Opérande HIST_ETAPE

`HIST_ETAPE = 'NON'`

Ce mot-clé permet de conserver tout l'historique des étapes/commandes utilisées. Ceci est gourmand en mémoire et ne doit être utilisé que pour des cas bien particuliers (les commandes qui le nécessitent l'indique dans leur documentation).
Par défaut, cet historique n'est pas conservé.

4.9 Mot-clé MESURE_TEMPS

Le mot clé MESURE_TEMPS permet de choisir le niveau de détail des impressions de temps CPU qui seront affichées dans le fichier de messages par les commandes effectuant des calculs élémentaires, des résolutions de systèmes linéaires, du déchargement d'objets sur disque ou des communications MPI.

4.9.1 Opérande NIVE_DETAIL

Par défaut, à la fin de chaque commande, on imprimera une ligne du type :

```
#1.Resolution.des.systemes.lineaires         CPU.(USER+SYST/SYST/ELAPS) :  7.52 0.79 11.22
#2.Calculs.elementaires.et.assemblages       CPU.(USER+SYST/SYST/ELAPS) : 15.07 0.70 15.77
```

◊ NIVE_DETAIL = 0 aucune impression.

= `1` impressions par défaut.
= `2` impressions plus détaillées :

```
#1.Resolution.des.systemes.lineaires          CPU (USER+SYST/SYST/ELAPS) :  7.72 0.82  8.72
#1.1.Numerotation,.connectivité.de.la.matrice CPU (USER+SYST/SYST/ELAPS) :  0.21 0.02  0.31
#1.2.Factorisation.symbolique                 CPU (USER+SYST/SYST/ELAPS) :  0.58 0.05  1.28
#1.3.Factorisation.numerique.(ou.precond.)    CPU (USER+SYST/SYST/ELAPS) :  6.78 0.73  7.71
#1.4.Resolution                               CPU (USER+SYST/SYST/ELAPS) :  0.15 0.02  0.35
#2.Calculs.elementaires.et.assemblages        CPU (USER+SYST/SYST/ELAPS) : 28.87 0.64 29.47
#2.1.Routine.calcul                           CPU (USER+SYST/SYST/ELAPS) : 26.61 0.56 26.61
#2.1.1.Routines.te00ij                        CPU (USER+SYST/SYST/ELAPS) : 24.58 0.07 25.78
#2.2.Assemblages                              CPU (USER+SYST/SYST/ELAPS) :  2.26 0.08  3.36
#2.2.1.Assemblage.matrices                    CPU (USER+SYST/SYST/ELAPS) :  2.02 0.06  3.12
#2.2.2.Assemblage.seconds.membres             CPU (USER+SYST/SYST/ELAPS) :  0.24 0.02  0.37
```

= `3` impressions plus détaillées et impression incrémentale pour chaque pas de temps.

Lors des calculs parallèles (MPI), le temps passé dans les communications est également affiché :

```
#4 Communications MPI                        CPU (USER+SYST/SYST/ELAPS)    : 12.67  0.50 12.68
```

4.9.2 Opérande MOYENNE

◊ `MOYENNE` = `'OUI'` affichage des statistiques (défaut)
= `'NON'` pas d'affichage des statistiques

Le mot-clé `MOYENNE` permet de contrôler l'affichage de statistiques supplémentaires exclusivement pour les calculs parallèles. Il s'agit de la moyenne des mesures sur tous les processeurs ainsi que l'écart-type de ces mesures.

Par défaut chaque temps affiché est complété ainsi :

```
#1 Résolution.des.systèmes.linéaires       CPU (USER+SYST/SYST/ELAPS) :  0.29  0.00  0.35
       (moyenne....diff..procs)            CPU (USER+SYST/SYST/ELAPS) :  0.30  0.00  0.47
       (écart-type.diff..procs)            CPU (USER+SYST/SYST/ELAPS) :  0.01  0.00  0.05
```

4.10 Mot-clé MEMOIRE

L'allocation des différentes structures de données est une allocation dynamique, l'utilisateur indique les limites de ressource lors du lancement de l'exécutable dans l'interface d'accès.

4.10.1 Opérande TAILLE_GROUP_ELEM

`TAILLE_GROUP_ELEM = tgrel [défaut: 1000]`

Ce paramètre donne le nombre maximum d'éléments finis d'un même type qui seront regroupés dans un groupe d'éléments.
Ce paramètre influence les performances mémoire et CPU des calculs élémentaires et des assemblages.
Quand on augmente `tgrel`, on doit en général gagner du temps CPU. En revanche, les objets JEVEUX sont plus gros, ce qui peut nécessiter plus de mémoire.

4.10.2 Opérande TAILLE_BLOC

`TAILLE_BLOC = tbloc [défaut: 800.]`

Ce paramètre donne la taille des blocs des matrices factorisées pour le solveur LDLT. Cette taille est donnée en kiloR8 (1 kiloR8 = 1024 réels). Ce paramètre influe sur le nombre d'opérations d'entrée / sortie et donc sur le temps d'assemblage et de résolution. Par défaut cette valeur est fixée à 800 kiloR8, soit 8 enregistrements par défaut sur le fichier d'accès direct associé à la base JEVEUX.

4.11 Mot-clé CATALOGUE

Ce mot clé est réservé aux développeurs, il est utilisé lors de l'opération de compilation des catalogues d 'éléments pour obtenir le fichier sous forme de base JEVEUX.

4.11.1 Opérande FICHIER

♦ FICHIER = nfic

Ne peut prendre que la valeur ' CATAELEM'

4.11.2 Opérande UNITE

◊ UNITE = unite

Numéro d'unité logique associée aux catalogues d'éléments. Dans les procédures de construction du catalogue d'éléments on utilise comme valeur 4. Le fichier fort.4 est obtenu à partir du contenu du répertoire des sources catalo à l 'aide d 'une procédure python.

4.12 Mot-clé RESERVE_CPU

Permet de réserver une part du temps CPU attribué au job pour terminer proprement l'exécution en cas d'arrêt par manque de temps CPU détecté par une commande Aster. Ce mécanisme n'est utile que dans le cas d'une exécution batch de *Code_Aster*. La valeur de cette réserve peut être indiquée en valeur absolue ou bien sous forme d'un pourcentage du temps CPU total. Cette valeur est bornée par la valeur du mot clé BORNE.
Lorsque le mot clé CODE est présent, c'est à dire pour l'ensemble des tests de non régression, on impose systématiquement une réserve de temps CPU de 10 secondes si le mot clé RESERVE_CPU est absent.

4.12.1 Opérande VALE

Valeur exprimée en secondes soustraite au temps CPU total, sur lequel certaines commandes globales se base pour arrêter proprement l'exécution.

4.12.2 Opérande POURCENTAGE

Pourcentage soustrait au temps CPU total, sur lequel certaines commandes globales se base pour arrêter proprement l'exécution.

4.12.3 Opérande BORNE

Valeur maximale de la réserve de temps, valant par défaut 180 secondes.